

Icon

The `<aava-icon>` component provides a comprehensive icon system built on top of Lucide icons, offering scalable vector graphics with extensive customization options, interaction support, and full accessibility compliance. It supports dynamic sizing, theming, interactive states, and seamless integration into any Angular application.

How to use

Basic Usage

Simple icon implementation using Lucide icon names with default styling.

```
import { AavaIconComponent } from "@aava/play-core";
```

Sizes

Flexible sizing options supporting both numeric pixel values and string-based sizing for responsive design.

```
&lt;!-- Basic Icon Usage - Angular Example --&gt;  
&lt;aava-icon iconName="home"&gt;&lt;/aava-icon&gt;  
&lt;aava-icon iconName="user"&gt;&lt;/aava-icon&gt;  
&lt;aava-icon iconName="heart"&gt;&lt;/aava-icon&gt;  
&lt;aava-icon iconName="star"&gt;&lt;/aava-icon&gt;  
&lt;aava-icon iconName="settings"&gt;&lt;/aava-icon&gt;
```

Size Options

- Numeric values - Direct pixel sizing (e.g., 16 , 24 , 32)
- String values - CSS-compatible units (e.g., '1rem' , '2em' , '100%')
- Responsive sizing - Use CSS custom properties for adaptive scaling

Colors

Comprehensive color customization with theme integration and semantic color support.

```
&lt;!-- Icon Sizes - Angular Example -->  
&lt;aava-icon iconName="star" iconSize="16"&gt;&lt;/aava-icon&gt;  
&lt;aava-icon iconName="star" iconSize="24"&gt;&lt;/aava-icon&gt;  
&lt;aava-icon iconName="star" iconSize="32"&gt;&lt;/aava-icon&gt;  
&lt;aava-icon iconName="star" iconSize="48"&gt;&lt;/aava-icon&gt;  
  
&lt;!-- String-based sizing -->  
  
&lt;aava-icon iconName="heart" iconSize="1rem"&gt;&lt;/aava-icon&gt;  
&lt;aava-icon iconName="heart" iconSize="1.5rem"&gt;&lt;/aava-icon&gt;  
&lt;aava-icon iconName="heart" iconSize="2rem"&gt;&lt;/aava-icon&gt;
```

Color System

- Custom colors - Direct hex, RGB, HSL, or named color values
- CSS custom properties - Theme-aware color tokens
- Semantic colors - Contextual colors based on component state
- Disabled state - Automatic color adaptation for disabled icons

Accessibility

Built-in accessibility features ensuring inclusive user experience and WCAG compliance.

```
&lt;!-- Icon Colors - Angular Example -->  
&lt;aava-icon iconName="palette" iconColor="#ff6b6b"&gt;&lt;/aava-icon&gt;  
&lt;aava-icon iconName="palette" iconColor="#4ecdc4"&gt;&lt;/aava-icon&gt;  
&lt;aava-icon iconName="palette" iconColor="#45b7d1"&gt;&lt;/aava-icon&gt;  
&lt;aava-icon iconName="palette" iconColor="#96ceb4"&gt;&lt;/aava-icon&gt;  
&lt;aava-icon iconName="palette" iconColor="#fffea7"&gt;&lt;/aava-icon&gt;
```

Accessibility Features

- Semantic HTML - Proper button element for interactive icons
- ARIA compliance - Appropriate ARIA attributes and roles
- Keyboard navigation - Full keyboard support for interactive icons
- Screen reader support - Meaningful descriptions and labels
- Focus management - Clear focus indicators and proper focus order
- High contrast - Support for high contrast accessibility modes

API Reference

Inputs

Property	Type	Default	Description
iconName	string	"	Lucide icon name to display
color	string	"	Custom color override (deprecated, use iconColor)
disabled	boolean	false	Whether the icon is disabled
iconColor	string	'#a1a1a1'	Icon color using hex, RGB, HSL, or CSS custom properties
iconSize	number string	24	Icon size in pixels (number) or CSS units (string)
cursor	boolean	false	Whether to show pointer cursor and enable interactions
hoverColor	string	'none'	Color applied on hover

Outputs

Event	Type	Description
userClick	EventEmitter<Event>	Emitted when interactive icon is clicked

Properties

Property	Type	Description
computedColor	string	Calculated color value based on state

Property	Type	Description
isInteractive	boolean	Whether icon accepts user interactions
isDisabled	boolean	Current disabled state

Methods

Method	Parameters	Description
handleClick()	event: Event	Process click events for interactive icons

CSS Custom Properties

Property	Description
--button-icon-color-disabled	Color used for disabled icon state
--icon-transition	Transition animation for state changes
--icon-hover-color	Color used on hover for interactive icons
--icon-focus-outline	Focus outline styling for accessibility
--icon-focus-outline-offset	Focus outline offset for clear visibility

Host Styling

The component applies the following host styles automatically:

Lucide Icon Integration

The component leverages Lucide icons, providing access to over 1,000 high-quality SVG icons.

Icon Library

- Comprehensive collection - 1,000+ carefully designed icons
- Consistent design - Uniform stroke width and styling
- Scalable vectors - Crisp appearance at any size

- Regular updates - Actively maintained icon library
- Tree-shakable - Only used icons are bundled

Icon Naming

Icons use kebab-case naming convention from the Lucide library:

Best Practices

Design Guidelines

- Choose appropriate sizes - Use consistent sizing across similar interface elements
- Maintain visual hierarchy - Larger icons for primary actions, smaller for secondary
- Use semantic colors - Colors should convey meaning and match design system
- Consider context - Icon choice should clearly communicate its purpose
- Consistent spacing - Maintain proper spacing around icons in layouts

Accessibility

- Provide meaningful context - Use proper ARIA labels for screen readers
- Interactive indication - Clearly distinguish interactive vs decorative icons
- Keyboard navigation - Ensure all interactive icons are keyboard accessible
- Focus indicators - Maintain clear focus states for navigation
- Color contrast - Ensure sufficient contrast for all color combinations

Performance

- Icon optimization - Lucide icons are automatically optimized SVGs
- Bundle efficiency - Tree-shaking ensures only used icons are bundled
- Avoid excessive sizing - Use reasonable icon sizes to maintain performance
- Minimize color changes - Frequent color updates can impact performance

Integration

- Theme consistency - Use CSS custom properties for consistent theming
- Component composition - Icons work well within buttons, forms, and navigation
- Event handling - Properly handle click events for interactive use cases
- State management - Consider disabled states in your application logic